UCRL-JC-140718

# Dataflow and Remapping for Wavelet Compression and Realtime View-Dependent Optimization of Billion-Triangle Isosurfaces

*M. A. Duchaineau, S. D. Porumbescu, M. Bertram B. Hamann, K. I. Joy*

This article was submitted to the National Science Foundation/Department of Energy – Lake Tahoe Workshop on Hierarchical Approximation and Geometrical Methods for Scientific Visualization

**U.S. Department of Energy**

Lawrence Livermore National Laboratory

**October 6, 2000**

# Dataflow and Remapping for Wavelet Compression and Realtime View-Dependent Optimization of Billion-Triangle Isosurfaces

Mark A. Duchaineau[1] Serban D. Porumbescu[1,3] Martin Bertram[2] Bernd Hamann[3] Kenneth I. Joy[3]

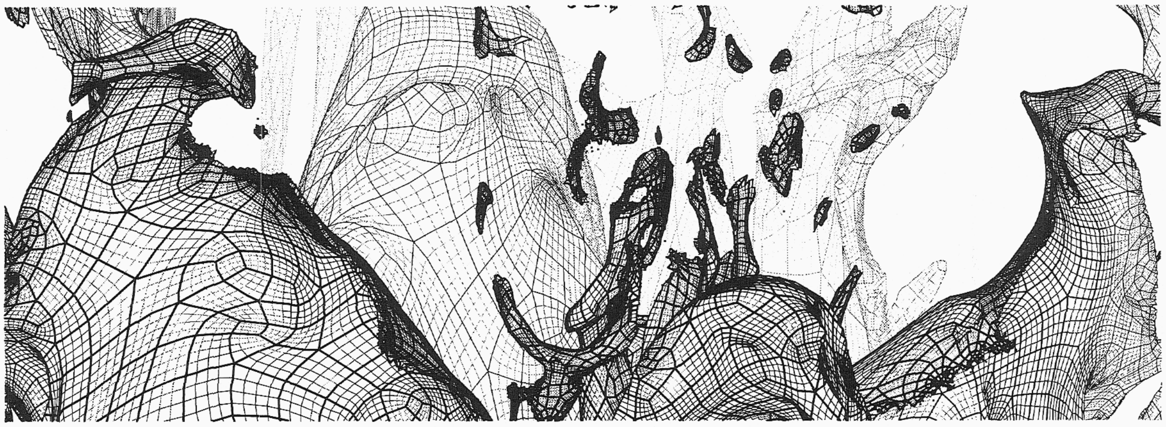

Figure 1: Shrink-wrap result for .3% of a terascale-simulation surface. This conversion from irregular mesh to a semi-regular form enables wavelet compression and view-dependent optimization.

## Abstract

Currently, large physics simulations produce 3D fields whose individual surfaces, after conventional extraction processes, contain upwards of hundreds of millions of triangles. Detailed interactive viewing of these surfaces requires powerful compression to minimize storage, and fast view-dependent optimization of display triangulations to drive high-performance graphics hardware. In this work we provide an overview of an end-to-end multiresolution dataflow strategy whose goal is to increase efficiencies in practice by several orders of magnitude. Given recent advancements in subdivision-surface wavelet compression and view-dependent optimization, we present algorithms here that provide the "glue" that makes this strategy hold together. Shrink-wrapping converts highly detailed unstructured surfaces of arbitrary topology to the semi-structured form needed for wavelet compression. Remapping to triangle bintrees minimizes disturbing "pops" during realtime display-triangulation optimization and provides effective selective-transmission compression for out-of-core and remote access to these huge surfaces.

## 1 Background

Terascale physics simulations are now producing tens of terabytes of output for a several-day run on the largest computer systems. An example: the Gordon Bell Prize-winning simulation of a Richtmyer-Meshkov instability in a shock-tube experiment [5] produced isosurfaces of the mixing interface with 460 million unstructured triangles using conventional extraction methods. New parallel systems are three times as large as when this run took place, so billion-triangle surfaces are to be expected shortly. Since we are interested in interaction, especially sliding through time, surfaces are precomputed (if they were not, the 100 kilotriangle-per-processor rates for the fastest isosurface extractors would result in several minutes per surface on 25 processors). Using 32-bit values for coordinates, normals and indices requires 16 gigabytes for a single surface, and several terabytes for a single surface tracking through all 274 time steps of the simulation. This already exceeds the compressed storage of the 3D fields from which the surfaces are derived, and adding additional isolevels or fields per time step would make this approach infeasible. With the gigabyte-per-second read rates of current RAID storage, it would take 16 seconds to read a single surface. A factor of 100 compression with no perceptible loss would cleanly solve both the storage and load-rate issues. This may be possible with new bicubic subdivision-surface wavelets [1].

Another bottleneck occurs with high-performance graphics hardware. The fastest commercial systems as of this writing can effectively draw around 20 million triangles per second, i.e. around 1 million triangles per frame at 20 frames per second. Thus almost a thousand-fold reduction in triangle count is needed. This level of reduction is too aggressive to be done without taking the interactively-changing viewpoint into account. As the scientist moves close to a feature of interest, that feature should immediately and seamlessly be fully resolved while staying within the interactive triangle budget. This can be formulated as the optimization of an adaptive triangulation of the surface to minimize a view-dependent error measure such as the projected geometric distortion on the screen. Since the viewpoint changes 20 times per second, and the error measure changes with the viewpoint, the million-element adaptation must be re-optimized continuously and quickly. The fastest theoretic time

[1] Center for Applied Scientific Computing (CASC), Lawrence Livermore National Laboratory, P.O. Box 808, L-561, Livermore, CA 94551, USA, duchaineau1@llnl.gov

[2] AG Computergraphik und Computergeometrie, Universitt Kaiserslautern, Postfach 3049, D-67653 Kaiserslautern, Germany, bertram@informatik.uni-kl.de

[3] Center for Image Processing and Integrated Computing (CIPIC), Department of Computer Science, University of California at Davis, Davis, CA 95616-8562, USA, {hamann, joy}@cs.ucdavis.edu

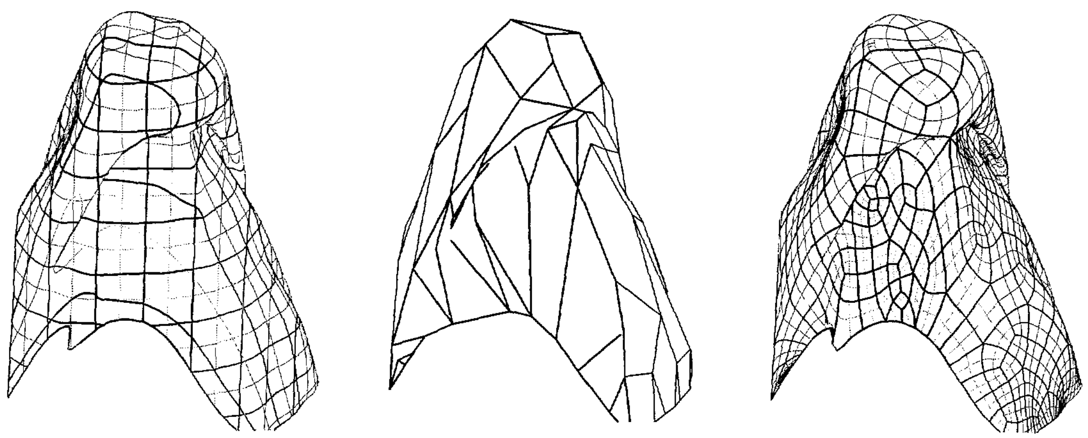

Figure 2: Shrink wrapping steps (left to right): the full-resolution isosurface, the base mesh constructed from edge collapses, and the final shrink-wrap with subdivision-surface connectivity.

that any algorithm can perform this optimization is $O(\Delta\text{output})$, an amount of time proportional to the number of element changes in the adaptation per frame. The ROAM algorithm achieves this optimal time using adaptations built from triangle bintrees [3]. With this approach in a flight-simulation example, around 3% of the elements change per frame, resulting in a thirty-times speedup in optimization rates. This is critically important since the bottleneck in fine-grained optimizers is processing time rather than graphics hardware rates.

The wavelet compression and view-dependent optimization are two powerful tools that are part of the larger dataflow from 3D simulation to interactive rendering. These are by no means the only challenging components of a terascale visualization system. For example, conversion is required to turn irregular extracted surfaces into a form appropriate for further processing (see Figure 1). Before turning to our focus on the two surface-remapping steps that tie together the overall dataflow, we give an overview of the complete data pipeline for surface interaction.

## 2 End-to-End Multiresolution Dataflow

The processing steps required to go from massive 3D field data to the interactively-changing optimal triangulations sent to graphics hardware involve six steps:

**Extract:** get unstructured triangles through accelerated isosurface extraction methods or through material-boundary extraction from volume-fraction data [2].

**Shrink-wrap:** convert the high-resolution unstructured triangulation to a similarly detailed surface that has subdivision-surface connectivity (i.e. is *semi-regular*), has high parametric quality, and minimizes the number of structured blocks. This uses three phases: (1) compute the signed-distance transform of the surface, (2) simplify the surface to a base mesh, and (3) iteratively subdivide, smooth and snap the new mesh to the fine-resolution surface until a specified tolerance is met.

**Wavelet compress:** for texture storage, geometry archiving, and initial transmission from the massively-parallel code runs, use high-order wavelets based on subdivision surfaces for nearly-lossless compression.

**Triangle bintree re-map:** re-map the shink-wrap parameterization to be optimal for subsequent view-dependent optimization (this is different than being optimal for high-quality wavelet compression). This involves piecewise-linear "wavelets" without any vanishing moments, but where most wavelet coefficients can be a single scalar value in a derived normal direction, and where highly localized changes are supported during selective refinement.

**Selective Decompress:** asynchronously feed a trickle of compressed detail where the view-dependent adaptation is most likely to find missing values during selective refinement in the near future. This trickle can be efficiently stored in chunks for efficient I/O or network transmission.

**Display-list optimization:** perform the realtime optimization of the display-list adaptive triangulation each frame, making maximal use of frame-to-frame coherence to accelerate frustum culling, element priority computation, and local refinement and coarsening to achieve the optimum per-view geometry.

The six processing steps occur in the order listed for terascale surface visualization. The first three steps—extraction, shrink-wrap and wavelet compress—typically occur in batch mode either as a co-process of the massively parallel physics simulation, or in a subsequent parallel post-processing phase. Given the selective access during interaction, and given the already existing wavelet hierarchy, the remapping for ROAM interaction can happen on demand given modest parallel resources. Because the ROAM remapping works in time $O(\text{output})$ in a coarse-to-fine progression, the amount of computation per minute of interaction is independent of the size of the physics grid or the fine-resolution surfaces. The ROAM remapper is envisioned as a runtime data service residing on a small farm of processors and disks, that reads, remaps and feeds a highly compact data stream on demand to the client ROAM display-list optimizer. This server-to-client link could be across the campus LAN or over high-speed WAN to provide remote access. the ROAM algorithm works at high speed per frame to optimize the display triangulation, and thus will reside proximate to the graphics hardware.

The remainder of this paper focuses on the two remapping steps in this dataflow. The first prepares a surface for wavelet compression, the second for ROAM triangle bintree optimization.

## 3 Shrink-Wrapping Large Isosurfaces

Before subdivision-surface wavelet compression can be applied to an isosurface, it must be re-mapped to a mesh with subdivision-surface connectivity. Minimizing the number of base-mesh elements increases the number of levels in the wavelet transform, thus

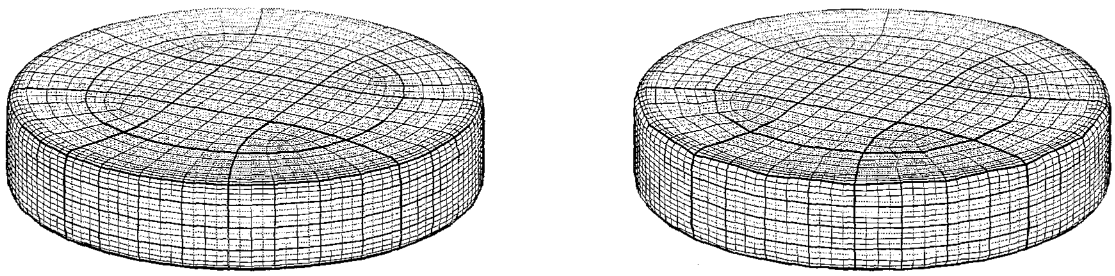

Figure 3: Before (left) and after the remapping for triangle bintree hierarchies. Tangential motion during subdivision is eliminated.

increasing the potential for compression. Because of the extreme size of the surfaces encountered, and the large number of them, the re-mapper must be fast, work in parallel, and be entirely automated. The compression using wavelets is improved by generating high-quality meshes during the re-map. For this, highly smooth and non-skewed parameterizations result in the smallest wavelet coefficient magnitudes, yielding small outputs after entropy coding. In addition, we would like to allow the new mesh to optionally have simplified topology, akin to actual physical shrink-wrapping of complex 3D objects with a few connected sheets of plastic.

The algorithm we propose for this shrink-wrapping is an elaboration of the method described in [1]. That method was used to demonstrate wavelet transforms of complex isosurfaces in a non-parallel, topology-preserving setting. The algorithm takes as input a scalar field on a 3D mesh and an isolevel, and provides a surface mesh with subdivision-surface connectivity as output, i.e. a collection of logically-square patches of size $(2^n + 1) \times (2^n + 1)$ connected on mutual edges. The algorithm at the high level is organized in three steps:

**Signed distance transform:** for each grid point in the 3D mesh, compute the signed-distance field, i.e. the distance to the closest surface point, negated if in the region of scalar field less than the isolevel. Starting with the vertices of the 3D mesh elements containing isosurface, the transform is computed using a kind of breadth-first propagation. Data parallelism is readily achieved by queueing up the propagating values on the boundary of a subdomain, and communicating to the block-face neighbor when no further local propagation is possible.

**Determine base mesh:** To preserve topology, edge-collapse simplification is used on the full-resolution isosurface extracted from the distance field using conventional techniques. This is followed by an edge-removal phase (edges but not vertices are deleted) that improves the vertex and face degrees to be as close to four as possible. Parallelism for this mode of simplification is problematic, since edge-collapse methods are inherently serial using a single priority queue to order the collapses.

To allow topology reduction, the 3D distance field is simplified before the isosurface is extracted. Simplification can be performed by using wavelet low-pass filtering on regular grids, or after resampling to a regular grid for curvilinear or unstructured 3D meshes. The use of the signed-distance field improves the simplified-surface quality compared to working directly from the original scalar field. To achieve the analog of physical shrink-wrapping, a max operation can be used in place of the wavelet filtering. This form of simplification is easily parallelized in a distributed setting.

**Subdivide and fit:** The base mesh is iteratively fit and the parameterization optimized by repeating three phases: (1) subdivide using Catmull-Clark rules, (2) perform edge-length-weighted Laplacian smoothing, and (3) snap the mesh vertices onto the original full-resolution surface with the help of the signed-distance field. Snapping involves a hunt for the nearest fine-resolution surface position that lies on a line passing through the mesh point in an estimated normal direction of the shrink-wrap mesh. The estimated normal is used, instead of e.g. the distance-field gradient, to help spread the shrink-wrap vertices evenly over high-curvature regions. The signed-distance field is used to provide Newton-Raphson-iteration convergence when the snap hunt is close to the original surface, and to eliminate nearest-position candidates whose gradients are not facing in the directional hemisphere centered on the estimated normal. Steps 2-3 may be repeated several times after each subdivision step to improve the quality of the parameterization and fit. In the case of topology simplification, portions of surface with no appropriate snap target are left at their minimal-energy position determined by the smoothing and the boundary conditions of those points that do snap. Distributed computation is straightforward since all operations are local and independent for a given level of resolution.

The shrink-wrap process is depicted in Figure 2 for approximately .016% of the 460 million-triangle Richtmyer-Meshkov mixing interface in topology-preserving mode. The original isosurface fragment contains 37,335 vertices, the base mesh 93 vertices, and the shrink-wrap result 75,777 vertices.

## 4 Re-Mapping for ROAM

The Realtime Optimally Adapting Meshes (ROAM) algorithm typically exploits a piecewise block-structured surface grid to provide efficient selective refinement for view-dependent optimization. A triangle bintree structure is used. This consists of a hierarchy of logically right-iscoceles triangles, paired across common base edges at a uniform level of subdivision. A simple split operation bisects the common base edge of such a pair, turning the two right-isosceles triangles into four. Merging reverses this operation. This is depicted in Figure 3.

The shrink-wrapping process that we have described produces meshes that are technically in this form, but cause large tangential motions of the mapping during refinement even in regions of flat geometry. To correct for this, we have devised a new remapping algorithm that eliminates tangential motion altogether whenever possible during ROAM refinement, but never causes the mapping to become degenerate or ill-defined. In effect, the surface is defined by a series of neighborhood height (normal) maps, allowing details to be stored with a single scalar rather than a 3-component displacement vector. Our method is similar to the independent work of Guskov *et al.* [4], differing primarily in the driving goal (efficient view-dependent optimization with crude wavelet compression in our case) and the details of mesh structure, subdivision scheme supported, intersection acceleration an so on.

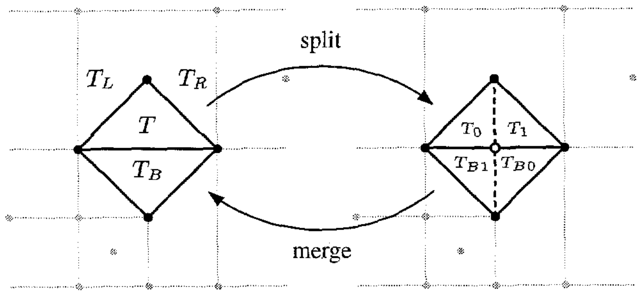


Figure 4: Split and merge operations on a bintree triangulation. A typical neighborhood is shown for triangle $T$ on the left.

The normal-remapping works from coarse to fine resolutions, remapping a complete uniform level of the hierarchy at once. The vertices of the base mesh are left fixed at their original positions. For every edge-bisection vertex formed in one subdivision step, estimated normals are computed by averaging the normals to the two triangles whose common edge is being bisected. For every vertex that has been remapped, its patch and parameter coordinates in the original map are kept. During edge bisection, the *parametric midpoint* is computed by topologically gluing at most two patches together from the original mesh, computing the mid-parameter values in this glued space, then converting those parameters back to unglued parameters. Given the constraints on our procedure, it is not possible for bisecting-edge endpoints to cross more than one patch boundary. A ray-trace intersection is performed from the midpoint of the line segment being bisected, in the estimated normal direction. Since we expect the intersection to be near the parametric midpoint in most cases, it is efficient to begin the ray intersection tests there for early candidates. Since the surface being ray-traced stays fixed throughout the remapping, the construction of typical ray-surface intersection-acceleration structures can be amortized and overall offer time savings (reducing the time from $O(N \log(N))$ to $O(N)$ for $N$ mesh vertices). Interval-Newton and finally Newton-Raphson iterations can be performed for the final precise intersection evaluation. Intersections are rejected if they are not within a parametric window defined by the four remapped vertices of the two triangles being split, shrunk by some factor (e.g. .5) around the parametric midpoint. The closest acceptable intersection is chosen. If none exist or are acceptable, the parametric midpoint is chosen.

The result of remapping is shown in Figure 4 for a test object produced by Catmull-Clark subdivision with semi-sharp features. The original parameterization on the left is optimal for compression by bicubic subdivision-surface wavelets, but produces extreme and unnecessary tangential motions during triangle-bintree refinement. The remapped surface, shown on the right, has bisection-midpoint displacements ("poor man's wavelets") of length zero in the flat regions of the disk, and displacements of minimized length elsewhere. We note that while the main motivation for this procedure is increasing accuracy and reducing the "pops" during realtime display-mesh optimization, the typical reduction to a single scalar value of the displacement vectors (wavelet coefficients) gives a fair amount of compression. This is desirable when the re-map from high-quality wavelet parameterization and compression is too time-consuming such as on the client end of the server-client asynchronous dataflow described earlier.

We note that the ROAM algorithm naturally requires only a tiny fraction of the current optimal display mesh to be updated each frame. Combined with caching and the compression potential of the remapping, this promises to provide an effective mechanism for out-of-core and remote access to the surfaces on demand during interaction.

# 5 Future Work

Several pieces of the terascale dataflow strategy have been realized to date, but many challenges remain to create a full capability:

1. For topology-preserving simplification, the inherently serial nature of the queue-based schemes must be overcome to harness parallelism.
2. Transparent textures or other means must be devised to handle the un-mapped surface regions resulting from topology-simplifying shrink wrapping.
3. The shrink-wrapping procedure can fail to produce one-to-one, onto mappings in some cases even when such mappings exist. Perhaps it is possible to revert to expensive simplification schemes that carry one-to-one, onto mappings only in problematic neighborhoods.
4. Shrink-wrapping needs to be extended to produce time-coherent mappings for time-dependent surfaces. This is a great challenge because of the complex evolution that surfaces go through during physics simulations.

# Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. We thank LLNL for support through the Science and Technology Education Program, the Student-Employee Graduate Research Fellowship Program, the Laboratory-Directed Research and Development Program, the Institute for Scientific Computing Research, and the Accelerated Strategic Computing Initiative.

# References

[1] Martin Bertram, Mark A. Duchaineau, Bernd Hamann, and Kenneth I. Joy. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization. *Proceedings of IEEE Vis00*, October 2000.

[2] Kathleen S. Bonnell, Daniel R. Schikore, Kenneth I. Joy, Mark Duchaineau, and Bernd Hamann. Constructing material interfaces from data sets with volume-fraction information. *Proceedings of IEEE Vis00*, October 2000.

[3] Mark A. Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. *IEEE Visualization '97*, pages 81–88, November 1997. ISBN 0-58113-011-2.

[4] Igor Guskov, Kiril Vidimce, Wim Sweldens, and Peter Schröder. Normal meshes. *Proceedings of SIGGRAPH 2000*, pages 95–102, July 2000.

[5] Arthur A. Mirin, Ron H. Cohen, Bruce C. Curtis, William P. Dannevik, Andris M. Dimits, Mark A. Duchaineau, Don E. Eliason, Daniel R. Schikore, Sarah E. Anderson, David H. Porter, Paul R. Woodward, L. J. Shieh, and Steve W. White. Very high resolution simulation of compressible turbulence on the IBM-SP system. *Supercomputing 99 Conference*, November 1999.